On How to Provision Quality of Service (QoS) for Large Dataset Transfers

Zhenzhen Yan, Malathi Veeraraghavan
Dept. of Electrical and Computer Engineering
University of Virginia
Charlottesville, VA, USA
Email: {zy4d,mvee}@virginia.edu

Chris Tracy, Chin Guok
Energy Sciences Network (ESnet)
LBNL
Berkeley, CA, USA
Email: {ctracy,chin}@es.net

***Abstract*—There is recent interest in using traffic-engineered, QoS-controlled paths for large-sized, high-rate dataset transfers in the scientific community. We refer to TCP flows created by such transfers as α flows. Research-and-education network providers are interested in intra-domain traffic engineering systems for identifying α flows at ingress routers within their networks, and redirecting them to traffic-engineered paths. This is primarily because of the adverse effects these α flows have on delay-sensitive multimedia flows. The focus of this work is to determine what QoS mechanisms are suitable to achieve the dual goals of preventing α flows from adversely affecting delay-sensitive flows, while simultaneously allowing them to enjoy high throughput. The interaction between policing schemes on the ingress interfaces and scheduling schemes on the egress interfaces was studied through a set of experiments on a high-speed router testbed. Our conclusions are that a scheduling-only mechanism, with no policing, is well suited to achieve these dual goals if the level of fairness offered by today's IP-routed service is sufficient for simultaneous α flows.**

***Keywords*—policing; scheduling; high-speed networks; traffic-engineering; virtual-circuit networks**

I. Introduction

For large-sized scientific dataset transfers, scientists typically invest in high-end computing systems that can source and sink data to/from their disk systems at high speeds. These transfers are referred to as α flows as they dominate other flows [1]. They also cause increased burstiness, which in turn impacts delay-sensitive real-time audio/video flows. In prior work [2], we proposed an overall architecture for an intra-domain traffic engineering system called Hybrid Network Traffic Engineering System (HNTES) that performs two tasks: (i) analyzes NetFlow reports offline to identify α flows, and (ii) configures the ingress routers for future α-flow redirection to traffic-engineered QoS-controlled paths. The prior paper [2] then focused on the first aspect, and analyzed NetFlow data obtained from live ESnet routers for the period May to Nov. 2011. The analysis showed that since α flows require high-end computing systems to source/sink data at high speeds, these systems are typically assigned static global public IP addresses, and repeated flows are observed between the same pairs of hosts. Therefore source and destination address prefixes of observed α flows can be used to configure firewall filter rules at ingress routers for future α-flow redirection. The effectiveness of such an offline α-flow identification scheme was evaluated with the collected NetFlow data and found to be 94%, i.e., a majority of bytes sent in bursts by α flows would have been successfully isolated had such a traffic engineering system been deployed [2].

The work presented here focuses on the second aspect of the HNTES by addressing the question of how to achieve α-flow redirection and isolation to traffic-engineered paths. Specifically, service providers such as ESnet [3] are interested in actively selecting traffic-engineered paths for α-flows, and using Quality-of-Service (QoS) mechanisms for policing and scheduling these flows. With virtual-circuit technologies, such as MultiProtocol Label Switching (MPLS), ESnet and other research and education network providers, such as Internet2, GEANT2, and JGN-X, offer a dynamic circuit service. An On-Demand Secure Circuits and Advance Reservation System (OSCARS) Inter-Domain Controller (IDC) [4] is used for circuit scheduling and provisioning. To support inter-domain (virtual) circuits, an IDC Protocol (IDCP) [5] is being standardized. The vircuit circuit (VC) setup phase offers an opportunity for path selection, and hence HNTES identifies the ingress/egress routers corresponding to the source and destination addresses of α flows, and requests intradomain circuits between these routers.

The basic interface to the IDC requires an application to specify the circuit rate, duration, start time, and the endpoints in its advance-reservation request. The specified rate is used both for (i) path computation in the call-admission/circuit-scheduling phase and (ii) policing traffic in the data plane. If the application requests a high rate for the circuit, the request could be rejected by the OSCARS IDC due to a lack of resources. On the other hand, if the request is for a relatively low rate (such as 1 Gbps), then the policing mechanism could become a limiting factor to the throughput of α flows, preventing TCP from increasing its sending rate.

The purpose of this paper is to evaluate the effects of different scheduling and policing mechanisms to achieve two goals: (i) reduction in delay and jitter of real-time sensitive flows that share the same interfaces as α flows, and (ii) support for high-throughput α-flow transfers.

Our *key findings* are as follows: **(i)** With the current widely deployed best-effort IP-routed service, which uses first-come-first-serve (FCFS) scheduling on egress interfaces of routers,

the presence of an α flow can increase the delay and jitter experienced by audio/video flows. **(ii)** This influence can be eliminated by configuring two virtual queues at the contending interface and redirecting identified α flows to one queue (an α queue), while all other flows are directed to a second queue (a β queue). **(iii)** If α flows use the dynamic circuit service offered by providers such as ESnet and Internet2, the currently configured policing mechanism will direct in-profile packets to a higher priority queue, and out-of-profile packets to a lower priority queue, which in turn, may have adverse effects on throughput. The reason of this degraded α-flow throughput is that the separation of in-profile and out-of-profile packets to different queues can cause out-of-sequence arrivals at the TCP receiver, which triggers TCP's fast retransmit/fast recovery congestion algorithm. **(iv)** An alternative approach to dealing with out-of-profile packets is to probabilistically drop a few packets using Weighted Random Early Detection (WRED), and to buffer the remaining out-of-profile packets in the same queue as the in-profile packets. This prevents the out-of-sequence problem and results in a smaller drop in α-flow throughput when compared to the separate-queues approach. Nevertheless, even with this WRED approach α-flow throughput is reduced when compared to the no-policing, scheduling-only solution. The WRED approach has a fairness advantage when multiple α flows are directed to the same α queue. However, preliminary NetFlow analysis indicates that the likelihood of two simultaneous α flows sharing a single link is fairly low if the α-flow threshold is relatively high (and it needs to be high in order to have adverse effects on other flows requiring its isolation). In summary, it may not be worth sacrificing α-flow throughput with policing if multiple simultaneous α flows occur rarely.

Section II provides background and reviews related work. Section III describes the experiments we conducted on a high-speed testbed to evaluate different combinations of QoS mechanisms and parameter values to achieve our dual goals of reduced delay/jitter for real-time flows and high throughput for α flows. Our conclusions are presented in Section IV.

II. Background and related work

The first three topics, historical perspective, a hybrid network traffic engineering system, and QoS support in state-of-the-art routers, provide the reader with relevant background information. The last topic, QoS mechanisms applied to TCP flows, covers related work.

Historical perspective: In the nineties, when Asynchronous Transfer Mode (ATM) [6] and Integrated Services (IntServ) [7] technologies were developed, virtual circuit (VC) services were considered for delay-sensitive multimedia flows. However, these solutions were not scalable to large numbers of flows because of the challenges in implementing QoS mechanisms such as policing and scheduling on a per-flow basis. Instead, a solution of overprovisioning connectionless IP networks to prevent buildups in router buffers was sufficient to meet delay requirements of real-time audio/video flows. While this solution works well most of the time, there are occasional periods when a single large dataset transfer is able to ramp up to a very high rate and adversely affect other traffic [8]. Such transfers, which are referred to as α flows, occur when the amount of data being moved is large, and the end-to-end sustained rate is high.

In the last ten years, there has been an emergent interest in using VCs but for α flow transfers not multimedia flows. As noted in Section I, service providers are interested in routing these α flows to traffic-engineered, QoS-controlled paths. The scalability issue is less of a problem here since the number of α flows is much smaller than of that of real-time audio-video flows. Based on the threshold chosen for α flows, this number could be as small as 1. It is interesting to observe this "flip" in the type of applications being considered for virtual-circuit services, i.e., from real-time multimedia flows to file-transfer flows.

Hybrid Network Traffic Engineering System (HNTES): Ideally if end-user applications such as GridFTP [9] alerted the provider networks en route between the source and destination before starting a high-rate, large-sized dataset transfer, these networks could perform path-selection and direct the resulting TCP flow(s) to traffic-engineered, QoS-controlled paths. However, most end-user applications do not have this capability, and furthermore inter-domain signaling to establish such paths requires significant standardization efforts. Meanwhile, providers have recognized that intra-domain traffic-engineering is sufficient if α flows can be automatically identified at the ingress routers. Deployment of such a traffic-engineering system lies within the control of individual provider networks, making it a more attractive solution. Therefore, the first step in our work was to determine whether such automatic α flow identification is feasible or not.

In our prior work [2], we started with hypothesis that computers capable of sourcing/sinking data at high rates are typically allocated static IP addresses, which means that the source-destination IP address prefixes can be used to identify α flows. If a NetFlow report for a flow showed that more than H bytes (set to 1 GB) were sent within a fixed time interval (set to 1 min), we classified the flow as an α flow, and stored the source and destination address prefixes (/24 and /32). This NetFlow data analysis is envisioned to be carried out offline on say a nightly basis for all ingress routers. If no flows are observed from a particular source-destination address prefix within an aging-out time period (set to 30 days), then the entry is removed. The effectiveness of this scheme was evaluated through an analysis of 7 months of NetFlow data obtained from an ESnet router. For this data set, 94% (82%) of bytes generated by α flows in bursts would have been identified correctly had /24 (/32) based prefix IDs been used. The results are consistent with findings from NetFlow data collected over 7 months from three other ESnet routers.

Given the effectiveness of this offline α-flow identification scheme, HNTES can provision firewall filters based on source/destination IP address prefixes to automatically detect

packets from α flows at a provider's ingress routers and redirect them to traffic-engineered, QoS-controlled paths.

QoS support in state-of-the-art routers: Multiple policing, scheduling and traffic shaping mechanisms have been implemented in today's routers. We review the particular mechanisms used in ESnet, and hence in our experiments. For scheduling, two mechanisms are used: Weighted Fair Queueing (WFQ) and Priority Queueing (PQ) [10]. With WFQ, multiple traffic classes are defined, and corresponding virtual queues are created on egress interfaces. Bandwidth and buffer space can be strictly partitioned or shared among the virtual queues. WFQ can be combined with PQ as explained later. On the ingress-side, policing is used to ensure that a flow does not exceed its assigned rate (used by the IDC during call admission). For example, in a single-rate two-color (token bucket) scheme, the average rate (which is the rate specified to the IDC in the circuit request) is set to equal the generation rate of tokens, and a maximum burst-size is used to limit the number of tokens in the bucket. The policer marks packets as *in-profile* or *out-of-profile*. Three different actions can be configured: (i) discard out-of-profile packets immediately, (ii) classify out-of-profile packets as belonging to a `Scavenger Service (SS)` class, and direct these packets to an `SS` virtual queue, or (iii) drop out-of-profile according to a WRED profile, but store remaining out-of-profile packets in the same queue as in-profile packets. For example, the drop rate for out-of-profile packets can increase linearly from 0 to 100 for corresponding levels of queue occupancy.

QoS mechanisms applied to TCP flows: Many QoS provisioning algorithms that involve some form of active queue management (AQM) have been studied [11]–[15]. Some of the simpler algorithms have been implemented in today's routers, such as RED [11] and WRED [13], while other algorithms, such as Approximate Fair Dropping (AFD) [15], have been shown to provide better fairness. An analysis of the configuration scripts used in core and edge routers of ESnet and Internet2 shows that these AQM related algorithms are not enabled. This is likely due to the commonly adopted policy of overprovisioning (a 2008 Internet2 memorandum [16] states a policy of operating links at 20% occupancy). Nevertheless, providers have recognized that in spite of the headroom, an occasional α flow can spike to a significant fraction of link capacity (e.g., our GridFTP log analysis showed transfers occurring at over 4 Gbps across paths of 10 Gbps links [8]), and that such spikes can adversely affect other flows. This explains the providers' interest in controlling the path taken by these flows, i.e., directing them to traffic-engineered QoS-controlled paths.

III. Experiments

A set of experiments are designed and executed to determine the best combination of QoS mechanisms with corresponding parameter settings in order to achieve our dual goals of reduced delay/jitter for real-time traffic and high throughput for α flows. For the first goal, we formulate a hypothesis as follows: a simple scheduling-only (no policing) scheme that isolates packets from α flows into a separate virtual queue on the egress interface from all other packets is sufficient to keep non-α flow delay/jitter low. *Experiment 1* tests this hypothesis.

In the current OSCARS IDC implementation, four classes-of-service (CoS) with corresponding virtual queues are used on the egress interfaces of routers: `network-control`, `best-effort`, `science-data`, and `scavenger-service`. The transmission rate and buffer allocation assigned to each of these queues is for example, 5%, 20%, 70%, and 5%, respectively. On the ingress side, policing is configured to check conformance of flows that requested circuits to their specified rates. In-profile packets are directed to the science-data queue, while out-of-profile packets are sent to the scavenger-service queue. We planned *experiment 2* to determine if this configuration of QoS mechanisms was suited to meeting our second goal of high-throughput for α flows. The expectation is that most circuit requests for file transfers will be for around 1 Gbps (on 10 Gbps links, this represents a significant fraction for just a single request), but as our prior work [8] showed scientific computing centers have hosts capable of sourcing/sinking data at over 4 Gbps. Policing such flows down to 1 Gbps will thus impact file-transfer throughput.

In *experiment 2*, out-of-profile packets resulting from ingress-side policing are directed to the scavenger-service (SS) queue, while in *experiment 3*, out-of-profile packets are subject to WRED as explained in Section II.

Section III-A describes the experimental setup, the experimental methodology, and certain router configurations that are common to all the experiments. Sections III-B, III-C, and III-D describe the three experiments, respectively.

A. Experimental Setup

The experimental network setup is shown in Fig. 1. The high-performance hosts, `W1` (West 1), `W2` (West 2), and `E1` (East 1), are Intel Xeon Nehalem E5530 models (2.4GHz CPU, 24GB memory) and run Linux version 2.6.33. The application hosts, `WA` (West App-host) and `EA` (East App-host), are Intel Dual 2.5GHz Xeon model and run Linux 2.6.18. The routers, `WR` (West Router) and `ER` (East Router), are Juniper MX80's running JunOS version 10.2. The link rates are 10 Gbps from the high-performance hosts to the routers, and 1 Gbps from the application hosts to the routers, and 10 Gbps between the routers.

This testbed is referred to as the Long Island MAN (LI-MAN), and is supported by ESnet as a DOE-funded testbed for networking research. The West-side hosts and routers are physically located in the Avenue-of-Americas (AoA) location in New York City, while the East-side hosts and routers are physically located in the Brookhaven National Laboratory (BNL) in Long Island, New York.

Each experiment consists of executing four steps: (i) plan the applications required to test a particular QoS mechanism, (ii) configure routers to execute the selected QoS mechanisms with corresponding parameter settings based on the planned

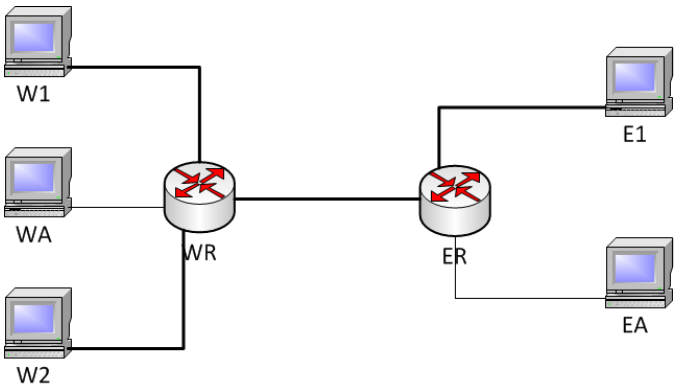

Figure 1. Experiment setup

application flows, (iii) execute applications on end hosts to create different types of flows through the routers, and (iv) obtain measurements for various characteristics, e.g., throughput, packet loss, and delay, from the end-host applications as well as from various counters in the routers.

A preliminary set of experiments were conducted to determine the specific manner in which the egress-side transmitter and buffer space were shared among multiple virtual queues. Theoretically both these resources can be strictly partitioned or shared in a work-conserving manner. If strictly partitioned, then even if there are no packets waiting in one virtual queue, the transmitter will not serve packets waiting in another queue. In this mode, each queue is served at the exact fractional rate assigned to it. In contrast, in the work-conserving mode the transmitter will serve additional packets from a virtual queue that is experiencing a higher arrival rate than its assigned rate if there are no packets to serve in the other virtual queues. Buffer space can similarly be shared in both modes. In all the experiments described below, the transmitter is shared among multiple virtual queues in work-conserving mode, while the buffer is shared in strictly partitioned mode.

Fig. 2 illustrates how a combination of QoS mechanisms was used in our experiments. *First*, incoming packets are classified into multiple classes based on pre-configured firewall filters, e.g., α-flow packets are identified by the source-destination IP address prefixes and classified into the α class. *Second*, packets in some of these classes are directly sent to corresponding egress-side virtual queues, while flows corresponding to other classes are subject to policing. A single-rate token bucket scheme is applied. If an arriving packet finds a token in the bucket, it is marked as being in-profile; otherwise it is marked as being out-of-profile. *Third*, for some policed flows, in-profile and out-of-profile packets are sent to separate egress-side virtual queues, while packets from other policed flows are subject to WRED before being buffered in a single virtual queue. On the egress-side, each virtual queue is assigned a priority level, a transmit rate (fraction of egress link capacity), and a buffer size. As noted in the previous paragraph the buffer allocation is strictly partitioned while the transmitter is shared in work-conserving mode. *Fourth*, the WFQ scheduler decides whether a virtual "queue is in-profile or not," by comparing the rate allocated to the queue and the rate at which packets have been served out of the queue. *Finally*, the PQ scheduler selects the queue from which to serve packets using their assigned priorities, but to avoid starvation of low-priority queues, as soon as a large enough number of packets are served from a high-priority queue to cause the status of the queue to transition to out-of-profile, the PQ scheduler switches to the next queue in the priority ordering. When all queues become out-of-profile, it starts serving packets again in priority order. It is interesting that while the *policer* is marking *packets* as in-profile or out-of-profile on a per-flow basis, the *WFQ scheduler* is marking *queues* as being in-profile or out-of-profile.

B. Experiment 1

1) Purpose and execution: The purpose of this experiment is to determine whether the simple scheduling-only solution of α-flow isolation to a separate virtual queue is sufficient to meet the first goal of keeping non-α flow delay/jitter low.

As per our execution methodology, the first step was to plan a set of applications. We decided to use three flows: a UDP flow, a high-speed TCP flow, and a "ping" flow. The application, `nuttcp`, is used to create both UDP and TCP flows. The UDP flow carries data from host `W2` toward host `W1`, while the TCP flow is from `E1` to `W1`. The TCP version used is H-TCP [17] because it is the best option to create high-speed (α) flows. The ping application sends repeated ICMP ECHO-REQUEST messages, one per second, from application host `EA` to high-performance host `W1`. Therefore, in this experiment, contention for buffer and transmitter resources occurs on the link from router `WR` to host `W1`. Although the high-performance host `W1` is the common receiver for all three flows, there is no CPU/memory resource contention at `W1` because the operating system automatically schedules the three receiving processes to three different cores.

The second step was to configure the routers. For comparison purposes, this experiment required two configurations: (i) `1-queue`: a single virtual queue is defined on the egress interface from `WR` to `W1`, and all three flows are directed to this queue, and (ii) `2-queues`: two virtual queues (α queue and β queue) are configured on the egress interface from `WR` to `W1`, and WFQ scheduling is enabled with the assigned transmitter rate (and buffer) percentages as follows: 60% for α queue and 40% for β queue. The priority of the α and β virtual queues was set to medium-high and medium-low, respectively. In the 2-queues configuration, two additional steps are required. A firewall filter is created in router `WR` to identify the TCP flow packets using its source and destination IP addresses (`E1` and `W1`, respectively). A class-of-service configuration command is used to classify these packets as belonging to the α class and to direct packets from this flow to the α queue on the egress interface from `WR` to `W1`. By default, all other packets are directed to the β queue, which means that packets from the UDP flow and ping flow will be directed to the β queue.

In the third step, the applications were executed as follows. Both the `nuttcp` UDP and ping flow were run for 200

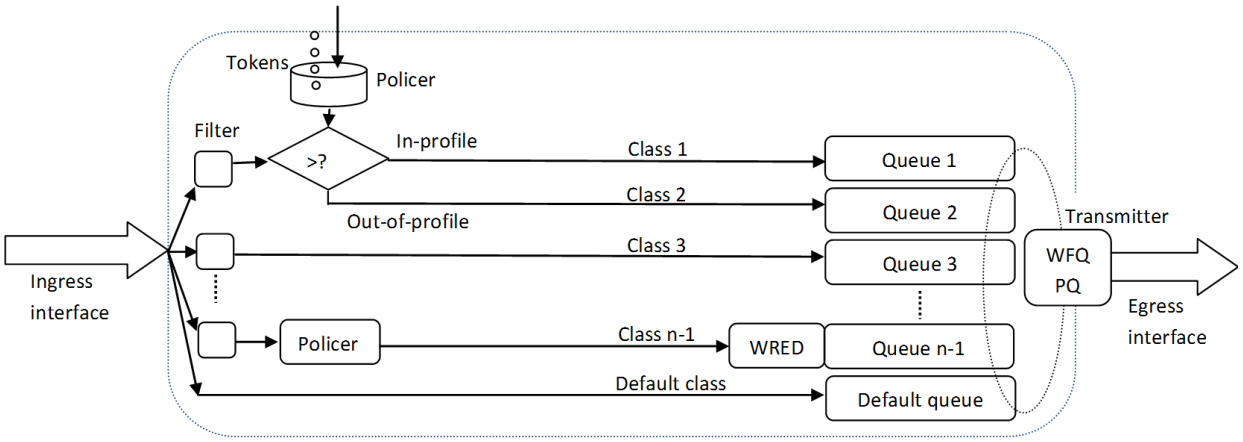

Figure 2. Illustration of QoS mechanisms in a router

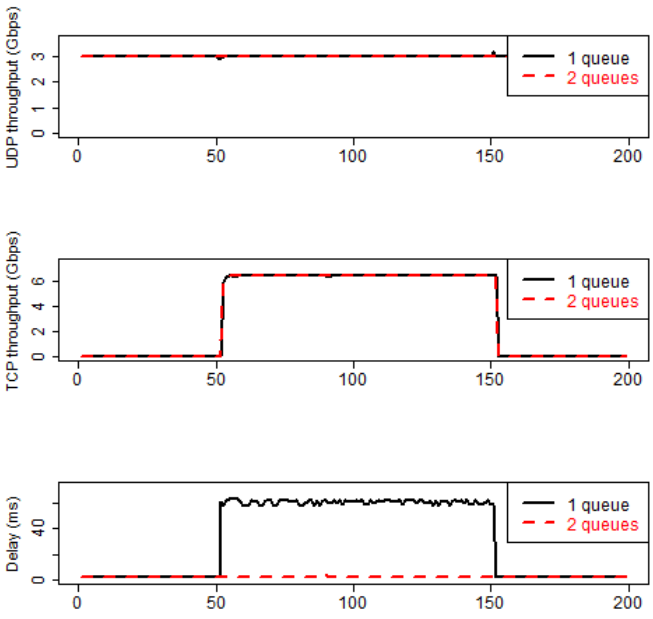

Figure 3. The x-axis is time measured in seconds; the top graph shows that the UDP rate is 3 Gbps in both the 1-queue and 2-queues configurations; the middle graph shows the TCP flow throughput; the bottom graph shows the delays experienced in the ping application.

seconds (from time 1 to time 200), while the `nuttcp` TCP flow was started at time 53 and run for 100 seconds. The rate of the UDP flow was set to 3 Gbps.

Finally, the UDP flow rate and TCP flow throughput reported in the next sub-section were obtained from measurements reported by the `nuttcp` application, and the ping delays were reported by the ping application.

2) Results and discussion: Fig. 3 illustrates that the simple scheduling-only solution of configuring two virtual queues on the shared egress interface and separating out the α flow packets into its own virtual queue leads to reduced packet delay/jitter for the β flows. In the 1-queue configuration, the mean ping delay across the 100 ping packets transmitted while the TCP flow was inactive was 2.28 ms and the standard deviation was 0.08 ms, while the mean and standard deviation of the 100 ping packets sent when the TCP flow was active were 60.6 ms and 1.64 ms, respectively. The ping delay increase is because the TCP (α) flow and the ping flow share the same single queue. In the 2-queues configuration, the mean and jitter of the ping delay were almost the same in the TCP-flow active and inactive periods. A small surge in ping delay to 4.5 ms occurred at time 91, which we ascertained was caused by network control packets exchanged between the routers.

Since the UDP flow rate at 3 Gbps was lower than the 40% assigned rate for the β queue, the latter was in-profile and hence the ping-application packets were served immediately, and not held up α-flow packets even though the α queue was sometimes out-of-profile. As explained in Section III-A, the PQ scheduler only honors priority if a queue is in-profile. It is interesting to note however that if the aggregate traffic directed to the β queue exceeds the β queue rate allocation when one or more α flows are present, then real-time flows could suffer from increased delay. Accurate estimation of the per-queue rate allocations is required.

C. Experiment 2

1) Purpose and execution: This experiment compares a 2-queues configuration (scheduling-only, no policing) with a 3-queues configuration (scheduling and policing), and furthermore compares multiple 3-queues configurations with different parameter settings.

As per our execution methodology, the first step was to plan applications. To study the behavior of the QoS mechanisms, the rate of the background traffic (an `nuttcp` UDP flow) was varied. Specifically, the same three application flows as in experiment 1 were planned, except that the rate of the `nuttcp` UDP flow was varied from 0 Gbps to 3 Gbps, and the `nuttcp` TCP flow was executed for the whole 200 sec.

In the second step, the router WR was configured with the following QoS mechanisms. The 2-queues configuration was the same as in experiment 1 (no policing), except that both queues were given equal weight in sharing the transmitter rate

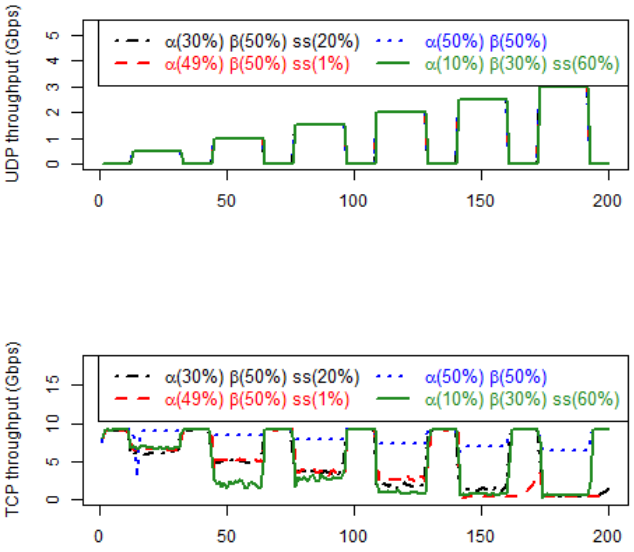

Figure 4. The x-axis is time measured in seconds; the top graph shows the on-off mode in which the UDP rate was varied; the lower graph shows the TCP flow throughput under the four configurations.

TABLE I. α-FLOW THROUGHPUT UNDER DIFFERENT BACKGROUND LOADS (UDP RATE) AND QOS CONFIGURATIONS

UDP rate (Gbps)	α-flow throughput (Gbps)			
	Percentages for 2-queues (α, β) and 3-queues (α, β, *SS*) configurations			
	(50,50)	(49,50,1)	(30,50,20)	(10,30,60)
0	9.12	9.09	9.07	9.12
0.5	8.92	6.62	6.06	6.83
1	8.43	5.22	5	2.12
1.5	7.94	3.78	3.67	2.82
2	7.44	2.7	1.93	0.92
2.5	6.95	0.33	1.38	0.69
3	6.46	0.34	0.38	0.61

and buffer space (50% each). For the 3-queues configurations, the percentages for the three queues (α, β, and *SS*) to which in-profile TCP-flow packets, UDP and ping packets, and out-of-profile TCP-flow packets, were directed, respectively, are shown in Table I. The priority of these three virtual queues is medium-high, medium-low, and low respectively. The policer is configured to direct in-profile TCP flow packets ($\leq$ 1 Gbps and burst-size $\leq$ 31 KB) to the α queue, and out-of-profile packets to the *SS* queue.

In the third step, experiment execution, the UDP flow rate was varied in a particular on-off pattern as seen in the top graph of Fig. 4. Finally, the performance metrics were collected as described for experiment 1.

2) Results and discussion: Fig. 4 shows the TCP throughput under the four configurations (one 2-queues and three 3-queues) for different rates of the background UDP flow. When the UDP flow rate is non-zero, since some of the plots overlap, we have summarized the mean TCP-flow throughput in Table I. When there is no background UDP traffic, the throughput of the TCP flow is around 9.1 Gbps for all four configurations as seen in the first row of Table I. As the background traffic load increases, the throughput of the TCP flow in all the 3-queues configurations drops more rapidly than in the 2-queues configuration, e.g., when the background UDP-flow rate is 3 Gbps, the TCP throughput is around 300-610 Mbps for the 3-queues configurations, while the TCP throughput is 6.5 Gbps for the 2-queues scenario (see last row of Table I).

In addition to explaining the first and last rows of Table I, we provide an explanation for the drop in TCP-flow throughput in the last column of the row corresponding to UDP rate of 1 Gbps, which highlights the importance of choosing the WFQ allocations carefully.

Explanation for the first row of Table I: The explanation for the TCP-flow throughput when there is no background traffic is straightforward in the 2-queues configuration. As there are no packets to be served from the β queue and the transmitter is operating in a working-conserving manner, the β queue's 50% allocation is used instead to serve the α queue, and correspondingly the TCP flow enjoys the full link capacity.

The explanation for the TCP-flow throughput values observed in the 3-queues configurations requires an understanding of the packet pattern incoming to the policer (see Fig. 2) and the rate at which packets leave the policer. When TCP-flow throughput is almost the line rate (over 9 Gbps), then the rate at which in-profile packets leave the policer will be almost constant at 1 Gbps. This is because the token generation rate is 1 Gbps and packet inter-arrival times are too short for a significant collection of tokens in the bucket. Therefore, it appears that in an almost periodic manner, every tenth packet of the TCP flow is marked as being in-profile and sent to the α queue and the remaining 9 packets are classified as out-of-profile and sent to the *SS* queue. Given that in all three 3-queues configurations, the WFQ scheduler will consider the α queue as being in profile (since even with the smallest allocation, this queue is assigned 10%), the PQ scheduler will systematically serve 1 packet from the α queue followed by 9 packets from the *SS* queue thus preserving the sequence of the TCP-flow packets. In the (49,50,1) configuration, 9 packets will be served out of the SS queue in sequence even though the queue would be regarded as out-of-profile after the first packet is served. This is because there are no packets in the β queue and none in the α queue given the policer's almost-periodic direction of 1-in-10 packets to this queue. Since no packets are out-of-sequence or lost, the TCP-flow throughput remains high at above 9 Gbps in all three 3-queues configurations.

Explanation for the last row of Table I: When there is background `nuttcp` UDP traffic at 3 Gbps, in the 2-queues configuration, it is easy to understand that the `nuttcp` TCP

flow is able to use up most of the remaining bandwidth, which is the line rate minus the rate of background `nuttcp` UDP flow, and hence the TCP-flow throughput is about 6.5 Gbps.

The explanation for the low `nuttcp` TCP throughput in the 3-queues configurations is that the opposite of the systematic behavior explained above for the first row occurs here. When the incoming packet rate to the policer is lower than the line rate, the token bucket has an opportunity to collect a few tokens. Therefore, when TCP-flow packets arrive at the policer, a burst of them will be classifed as in-profile (since for every token present in the bucket, one packet is regarded as being in-profile), and sent to the α queue. These will be served in sequence, but because the transmitter has to serve the β queue (for the UDP flow), the pattern in which the policer sends packets to the α queue and *SS* queue is more unpredictable and involves bursts. This results in TCP segments arriving out-of-sequence at the receiver (as confirmed with `tcpdump` and `tcptrace` analyses presented in the next section). Out-of-sequence arrivals triggers TCP's Fast retransmit/Fast recovery algorithm, which causes the sender's congestion window to halve resulting in lower throughput.

Explanation for the last-column entry in the row corresponding to 1 Gbps in Table I: The TCP-flow throughput drops much faster from 6+ Gbps to 2.12 Gbps when UDP rate increases from 0.5 to 1 Gbps in the (10,30,60) 3-queues configuration than in the other two 3-queues configurations. This is explained using the above-stated reasoning that when the TCP-flow packets do not arrive at close to the line rate, the inter-packet arrival gaps allow the token bucket to collect a few tokens, making the policer send bursts of packets to the α queue. In this (10,30,60) configuration, after serving only one packet from each burst, the WFQ scheduler will declare the α queue to be out-of-profile since its allocation is only 10% or equivalently 1 Gbps. This will lead to a greater number of out-of-sequence arrivals at the TCP receiver than in the other two 3-queues configurations, and hence lower throughput.

In summary, the higher the background traffic load, the lower the `nuttcp` TCP-flow packet arrival rate to the policer, the larger the inter-arrival gaps, the higher the number of collected tokens in the bucket, and the larger the number of in-profile packets directed to the α queue. If the WFQ allocation to the α queue is insufficient to serve these in-profile bursts, packets from the α queue and *SS* queue will be intermingled resulting in out-of-sequence packets at the receiver. This fine point notwithstanding, the option of directing out-of-profile packets from the policer to a separate queue appears to be detrimental to α-flow throughput. We conclude that the second goal of high α-flow throughput cannot be met with this policing approach. In the next experiment, a different mechanism of dealing with out-of-profile packets is tested.

D. Experiment 3

1) Purpose and execution: This experiment compares the approach of applying WRED to out-of-profile packets rather than redirecting these packets to a scavenger-service queue as in experiment 2. As per our execution methodology, the planned applications are the same as in experiment 1. The UDP-flow rate is maintained at 3 Gbps throughout the 200 sec time interval.

TABLE II. QOS CONFIGURATIONS FOR EXPERIMENT 3

Configuration	Policing	WFQ allocation `2-queues`:(α,β) `3-queues`:(α,β,`SS`)	WRED
`2-queues`	None	(60,40)	NA
`3-queues + policing1`	OOP to `SS` queue	(59,40,1)	NA
`3-queues + policing2`	OOP to `SS` queue	(20,40,40)	NA
`2-queues + policing + WRED`	WRED	(60,40)	Drop prob. = queue occ.

The next step is router configuration. Four configurations are compared as shown in Table II. OOP stands for Out-of-Profile packets. In the fourth option, OOP packets are dropped probabilistically at the same rate as the fraction of α-queue occupancy. In other words, if the α queue has 50% occupancy, then 50% of the OOP packets are dropped on average.

The applications were executed to generate one `nuttcp` TCP flow and one `nuttcp` UDP flow. Finally, in addition to the previously used methods of obtaining throughput reports from `nuttcp`, two packet analysis tools, `tcpdump` and `tcptrace`, were used to determine the number of out-of-sequence packets at the receiver. Additionally, to find the number of lost packets, a counter was read at router `WR` for the `WR`-to-`W1` link before and after each application run.

2) Results and discussion: The lower graph in Fig. 5 and Table III show that the TCP-flow throughput is highest in the 2-queues (no policing) scenario, with the WRED option close behind. The policing with WRED option performs much better than the options in which out-of-profile (OOP) packets are directed to an *SS* queue. In the WRED-enabled configuration, the TCP flow experiences a small rate of random packet loss, as shown in Table III, while in redirect-OOP-packets-to-*SS*-queue configurations, there are much higher numbers of out-of-sequence packets. The out-of-sequence packets in the WRED-enabled configuration result from the 15 lost packets, and are not independent events.

Surprisingly, even though the number of out-of-sequence packets is larger for the `3-queues + policing1` configuration, the throughput is higher in that configuration. This implies that a fewer number of the out-of-sequence packets caused triple-duplicate ACKs in the first case. But this pattern is likely to change for repeated executions of the experiment.

Finally, Fig. 5 shows that in the `2-queues` (no policing) configuration, there is degradation of throughput soon after the flow starts. Also, Table III shows a loss of 5050 packets (the 4076 out-of-sequence packets were related to these losses) from tcptrace, we found that these losses occur at the start of the transfer. This is explained by the aggressive growth

TABLE III. NUMBER OF OUT-OF-SEQUENCE PACKETS AND LOST PACKETS FOR DIFFERENT QOS SETTINGS

Measure	2queues	3queues+ policing1	3queues+ policing2	2queues+ policing+wred
Average throughput	6 Gbps	0.92 Gbps	0.47 Gbps	5.6 Gbps
Num. of out-of-sequence packets at the receiver	4076	8812	7199	15
Num. of lost packets at the WR-to-W1 router link	5050	0	0	15

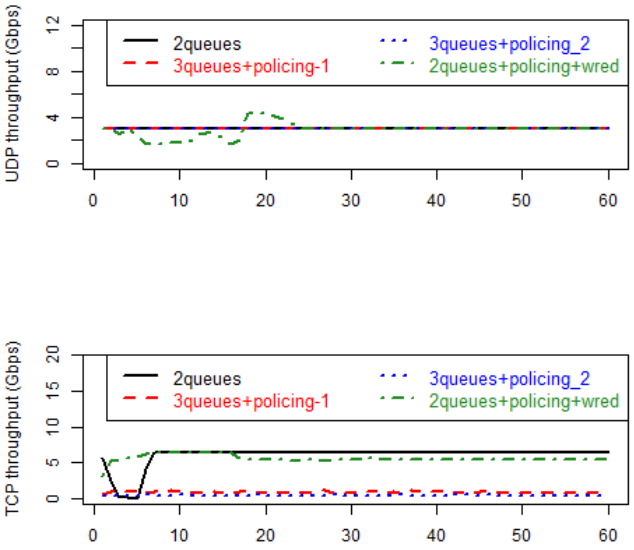

Figure 5. The x-axis is time measured in seconds; the top graph shows the on-off mode in which the UDP rate was varied; the lower graph shows the TCP flow throughput under the four configurations.

of the congestion window (cwnd) in H-TCP, which uses a short throughput probing phase at the start. During the 1st second, the throughput of the nuttcp TCP flow averaged 5.7 Gbps. The 5050 lost packets occurred in the 2^{nd} second. These losses occurred in the WR router buffer on its egress link from WR to W1. If H-TCP increased its cwnd to a large enough value to send packets at an instantaneous rate higher than 7 Gbps, then given the presence of the UDP flow at 3 Gbps, the α queue would fill up. Through experimentation, we determined that the particular router used as WR has a 125 MB buffer. Since the buffer is shared between the α and β queues in a strictly partitioned mode with the 60-40 allocation, the α queue has 75 MB, which means that if the H-TCP sender exceeds the 7 Gbps rate by even 600 Mbps, the α queue will fill up within a second. Inspite of this initial packet loss, the 2-queues no-policing configuration achieves the highest throughput. The 2-queues+policing+WRED configuration will likely be more fair if multiple α flows are directed to the same α queue. For example, the AFD approach [15] offers a dropping mechanism to achieve fairness between TCP flows. In the 2-queues no-policing configuration, α flows will experience the same fairness level as in today's best-effort network, achieve high-throughput while simultaneously not impacting the delay/jitter of real-time flows. A preliminary analysis of ESnet NetFlow data shows that when the defining threshold for α flows is relatively high, it is only on rare occasions that multiple α flows from different transfers share the same link (some transfers use multiple parallel TCP flows as observed in our GridFTP log analysis [8]).

IV. Conclusions

This paper presented an approach to QoS provisioning for α flows (high-rate, large-sized file transfers) for two purposes: (i) to reduce the adverse effects they can cause on delay-sensitive flows, and (ii) to maximize the throughput of α flows. Several experiments were conducted to compare different QoS mechanisms on state-of-the-art routers. We showed that a simple 2-queue scheme in which α flows are isolated to their own queue is sufficient to achieve the first goal. As for the second goal, we investigated the effects of two policing schemes. A scheme that is commonly deployed in research-and-education networks (REN) separates out in-profile and out-of-profile packets from an α flow into two different virtual queues. The policed rate is determined by the rate requested during circuit setup (REN providers offer a dynamic circuit service that is used by α flows). However, it is difficult to accurately gauge the rate at which a file transfer can be executed, and sometimes α flows exceed their requested rates. When this happens, the solution of using two queues causes a significant number of out-of-sequence packets at the receiver, and TCP's fast retransmit/fast recovery method reduces throughput. An alternative approach is to use Weighted Random Early Detection (WRED) and drop out-of-profile packets probabilistically, but keep the remaining out-of-profile and in-profile packets in the same queue. This mechanism results in higher throughput than the deployed approach, but it nevertheless reduces α-flow throughput. Therefore, to meet our dual goals, we recommend a scheduling-only, no-policing approach.

V. Acknowledgment

The University of Virginia portion of this work was supported by the U.S. Department of Energy (DOE) grant DE-SC0002350, DE-SC0007341 and NSF grants OCI-1038058, OCI-1127340, and CNS-1116081. The ESnet portion of this work was supported by the Director, Office of Science, Office of Basic Energy Sciences, of the U.S. DOE under Contract No. DE-AC02- 05CH11231. This research used resources of the ESnet ANI Testbed, which is supported by the Office of Science of the U.S. DOE under contract DE-AC02-05CH11231, funded through the American Recovery and Reinvestment Act of 2009.

References

[1] S. Sarvotham, R. Riedi, and R. Baraniuk, "Connection-level analysis and modeling of nework traffic," ACM SIGCOMM Internet Measurement Workshop 2001, Nov. 2001, pp. 99-104.

[2] Z. Yan, C. Tracy, and M. Veeraraghavan, "A hybrid network traffic engineering system," Proc. of IEEE 13th High Performance Switching and Routing (HPSR) 2012, Jun. 24-27, 2012, pp. 141-146.

[3] Esnet. Retrieved: 02.11.2013. [Online]. Available: http://www.es.net/

[4] On-Demand Secure Circuits and Advance Reservation System (OSCARS). Retrieved: 02.11.2013. [Online]. Available: http://www.es.net/services/virtual-circuits-oscars

[5] (2010, Feb.) Inter-domain Controller (IDC) Protocol Specification. Retrieved: 02.11.2013. [Online]. Available: http://www.controlplane.net/

[6] J. Spragins, "Asynchronous Transfer Mode: Solution for Broadband ISDN, Third Edition [New Books]," Jan./Feb. 1996, pp. 7.

[7] E. R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP)," RFC 2205, Sep. 1997.

[8] Z. Liu, M. Veeraraghavan, Z. Yan, C. Tracy, J. Tie, I. Foster, J. Dennis, J. Hick, Y. Li, and W. Yang, "On using virtual circuits for GridFTP transfers," The International Conference for High Performance Computing, Networking, Storage and Analysis 2012 (SC 2012), Nov. 10-16, 2012.

[9] GridFTP. Retrieved: 02.11.2013. [Online]. Available: http://globus.org/toolkit/docs/3.2/gridftp/

[10] J. Kurose and K. Ross, "Computer networks: A top down approach featuring the internet," Pearson Addison Wesley, 2010.

[11] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," IEEE/ACM Transactions on Networking, Aug. 1993, pp. 397-413.

[12] D. Lin and R. Morris, "Dynamics of random early detection," ACM SIGCOMM Computer Communication Review, 1997, pp. 127-137.

[13] WRED. Retrieved: 02.11.2013. [Online]. Available: http://www.cisco.com/en/US/docs/ios/11_2/feature/guide/wred_gs.html

[14] R. Guérin, S. Kamat, V. Peris, and R. Rajan, "Scalable QoS provision through buffer management," vol. 28, no. 4, 1998, pp. 29-40.

[15] R. Pan, L. Breslau, B. Prabhakar, and S. Shenker, "Approximate fairness through differential dropping," ACM SIGCOMM Computer Communication Review, pp. 23–39, 2003.

[16] R. P. Vietzke. (2008, Aug.) Internet2 headroom practice. Retrieved: 02.11.2013. [Online]. Available: https://wiki.internet2.edu/confluence/download/attachments/17383/Internet2+Headroom+Practice+8-14-08.pdf

[17] D. Leith and R. Shorten, "H-TCP: TCP for high-speed and long-distance networks," Proc. of PFLDnet, Feb. 16-17, 2004.